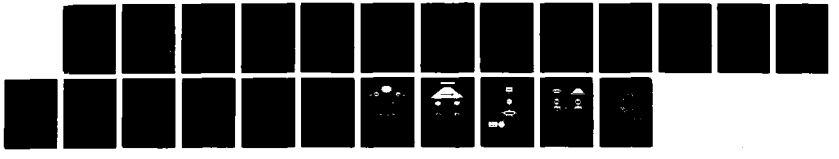
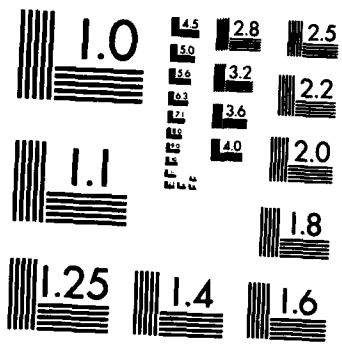
AD-A188 150 SHARED WORKSPACES FOR REAL-TIME COLLABORATION IN DISTRIBUTED NETWORKS: CO..(U) NORTH CAROLINA UNIV AT CHAPEL HILL DEPT OF COMPUTER SCIENCE 1/1

UNCLASSIFIED H M ABDEL-WAHAB ET AL. 1987 F/G 25/5 NL





Contract N00014-86-K-0680

Shared Workspaces for Real-time Collaboration in Distributed Networks: Concepts, Techniques, Problems

Hussein M. Abdel-Wahab
Department of Computer Science
North Carolina State University
Raleigh, NC 27695

Sheng-Uei Guan and J. Nievergelt
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27514

Abstract

Until recently, computer-based collaboration between geographically dispersed users has been limited primarily to electronic mail and file transfer. There is an increasing interest in computer support for real-time interaction between such users. The main goal of our research is to enable users to remotely share and operate simultaneously on "objects" contained in a "workspace". Examples of objects are textfiles, graphs, or images. All participants have identical views of the objects. Users manipulate and operate on the objects using appropriate "tools". For example, an appropriate tool for operating on a textfile is a text editor. The basic entities of our model: users, objects and tools, may reside at arbitrary sites of a computer network. Users who wish to collaborate and share a workspace will form a "session". For each session, a process called the "session server" regulates access to workspace objects efficiently and fairly. All session servers of a given machine are interfaced to a process called the "communications server". The communications server is responsible for connecting users and moving objects between different machines. We have implemented a prototype system of remote shared workspaces where objects are restricted to be textfiles, and tools can only accept input from keyboards. Our approach differs from other collaborative tools in that we offer a general purpose utility that converts any single-user tool into one that can be used for real-time collaboration among several remote users.

KEY WORDS: Distributed systems, Real-time collaboration, UNIX, Interprocess communications

1 Introduction

Today, computer-based collaboration between geographically dispersed users is still limited primarily to electronic mail and file transfer, but several research groups [1, 2, 3] experiment with more powerful computer support for team work. Let us summarize the features of some of the prototypes described in the literature.

At the MIT Laboratory of Computer Science, users share information from their personal calendars to schedule meetings using a real-time conferencing system, RTCAL[4]. Participants speak to each other over the phone and use their workstation display as a blackboard. Another system, CES, is a collaborative editing system for co-authors working asynchronously on a shared structured document, where users can work independently on separate sections of the document[5].

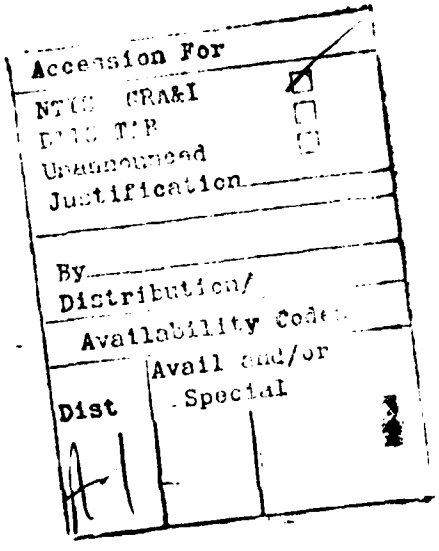
At the Xerox PARC Intelligent Systems Laboratory, an experimental meeting room, Colab, provides computational support for collaboration in face-to-face meetings. Several prototype collaboration systems have been built using Colab, such as WYSIWIS (What you see is what I see) which is a multiuser interface that expresses many of the characteristics of a chalkboard in face to face meetings[6,7].

At IBM, an asynchronous conference system has been implemented for Bitnet and Vnet[8]. It is not based on real-time interaction, users send and receive information at their own convenience. At Stanford, an experimental multimedia conferencing facility has been implemented using the V-system, a message-based distributed operating system[9].

In summary, research on computer support for people working together simul-

taneously, where real-time interaction is essential, is still in its beginnings. Most work is based on experimental systems that exist only in one laboratory. In contrast to the prevailing trend, our objective is to build usable, low-cost remotely shared workspaces based on widely available systems and single-user applications programs, in order to allow a large user community to experiment with this new technology of real-time collaboration. With the proliferation of high bandwidth computer networks, and the increasing popularity and affordability of powerful workstations, it is feasible to provide the users with the environment to achieve this goal.

In Section 2, we present the major components of our system and the overall system architecture. A simple prototype that we have implemented in C under 4.3BSD UNIX is discussed in Section 3. We conclude by outlining our future research goals and directions.



2 System Overview and Architecture

The system to be described is intended to link institutions, such as universities or industrial research organizations, that have most of their computers connected to local area networks (LANs). The LANs of nearby institutions are usually interconnected by high bandwidth links using, for example, microwave transmission, such as the MCNC communications network in North Carolina's Research Triangle (Fig. 1). LAN's of distant institutions are typically interconnected via moderate bandwidth long haul networks such as ARPANET or BITNET. The main objective of our research is to bring users of those remote systems together to collaborate in reviewing and editing documents containing text, graph and image objects. In this paper we use the term "Workspace" to denote a collection of such objects belonging to some application, and the software needed to access these objects. For example, researchers writing a joint paper will create objects such as sections, figures and tables in their workspace; whereas a group of physicians reviewing a patient case might work on lab results, charts and X-ray images. For each type of object there is a set of appropriate "tools" that can be used to manipulate and operate on the objects, e.g., for textual objects, the set of tools includes text editors, formatters and spelling checkers.

Users collaborating in a shared workspace are said to form a "session", and one user is identified as the "session chairman". The chairman manages the session, for example, by invoking an appropriate tool and admitting a new participant to the session. For each session, a process called the "session server " (abbreviated as

SS) acts as the interface between users and the tool. Each host in the network has a single process called the "communications server" (abbreviated as CS) which is responsible for all intermachine communications, it is a demon process created at boot time. Figure 2 depicts the general system architecture and the relationship between the different components of the model.

2.1 Creating New Sessions

To create a session, a user (who subsequently becomes the session chairman) asks the communications server CS on his machine to create a new SS process. Once process SS is created, it asks the user to provide values for the following session "parameters":

1. Initial list of participants: Give a list of all users who may join the session. Each user on the list is specified as:

   user-id@machine-id

   In UNIX, a *user-id* is his login name, and the *machine-id* is the DARPA Internet machine name or address. For example,

   `wahab@dopey.cs.unc.edu` or `wahab@128.109.136.82`

   If the *machine-id* is not specified, the local host is assumed.

2. Initial list of workspace objects: Give the list of objects to be included initially in the workspace. An object is identified by:

   object-id@machine-id

   For example, under UNIX an *object-id* is a path name such as:

Also, if the *machine-id* is not specified, the local host is assumed.

3. Session mode: There are three basic modes: *open, closed* and *secret.*

   - *open:* In this mode, it is possible for a user who is not on the initial list to join an ongoing session provided he obtains the approval of the session chairman.
   - *closed:* The session is restricted only to participants on the initial list and no one can join the session once it has started. This may be useful in situation when participants wish to work without disturbance.
   - *secret:* In addition to the restrictions of the closed mode, the system does not reveal any information about the existence of the session, as if it never existed. This may be useful in situations where sensitive objects are manipulated such as in preparing examinations and evaluation reports.

We allow an *observer* option for open and closed mode, but not for secret mode. Subject to the approval of the session chairman, an observer may join a session in "read-only mode".

2.2 Joining a Session

Following the specification of the session parameters, the process SS sends a message to all users in the participants list inviting them to join the session. The

message contains a session identification of the form:

session-id@machine-id

where the *session-id* is an integer that uniquely identifies the session on the machine. For example:

`534@dopey.cs.unc.edu`

To join the session, a participant asks the communication server CS on his machine to connect him to the specified session. Process CS creates a "user agent" process and two "windows", as shown in Figure. 3. Window W_t is used for tool input and output, window W_c for session control and for conversation between participants.

2.3 Starting a Session

After all expected users have joined the session, or the chairman has decided that a sufficient number of participants have joined the session and it is time to start, he issues a "start session" command. The session server SS sends a message to every participant signaling the start of the session. Late users can join the session at any time, at the discretion of the chairman.

2.4 Invoking Tools During Session

When the session starts, the chairman invokes any appropriate tool. The tool is specified as:

tool-id@machine-id

For example in UNIX, a *tool-id* is the path name of the program e.g., */bin/vi*. If

the *machine-id* is not specified, the tool is assumed to be available in the local machine. If the tool is local, then the session server SS is directly connected to the tool process. If the tool is available on a remote machine, the session server SS asks the communications server CS of the remote machine to facilitate the execution of the tool, as follows: First, the needed objects are copied to the remote site, since most tools only operate on local objects; to copy the required objects, both the session server and the remote communication server execute an "object replication process" (see Figure 4). Second, the tool process starts and its input/output is connected to the session server through the "tool agent " process as shown in Figure. 3. Whenever a user exits from his tool, the modified and the newly created objects are moved back into the workspace at the session server site. Note that any participant, not necessarily the chairman, can end the use of the tool, since the session server does not interpret tool commands and hence has no control over who may terminate the tool.

2.5 Tool Control and Use

If more than one user simultaneously issues tool commands, this may lead to serious problems and cause undesirable results. For example, if two users attempt to delete the same line of a text file, two lines instead of one may be deleted, since the editor accepts both commands. We solved this problem by introducing a "token": only the user holding the token (the "active user") is able to provide input to the tool. The user requests the token by entering a "get token" command

in window W_c, and may release it with a "release token" command. The session server circulates the token among users fairly and efficiently. For effective work, the active user must be guaranteed an uninterrupted quantum of time, q, once he gets the token. For fairness, other participants must be able to request the token and obtain it within a certain known waiting time. When the active user has to release the token, he is given a brief grace period, g, to complete his current task. Values for q and g can be set when a tool is started, depending on the tool and the number of users. Tool commands are entered in the tool window W_t as usual, exactly as described in the tool manual.

2.6 Conversation During a Session

Users can talk to each other using a telephone conference call, or they can use the control window W_c as a "chalkboard" for posting messages. Any line typed in window W_c which does not correspond to a session control command, is broadcast to all users preceded by the name of the writer. Observers are also allowed to send such messages, and thus can interject question and comments. Observers cannot, however, get the token and provide input to the tool; thus they cannot modify the objects.

2.7 Leaving and Ending a Session

A user may leave the session at any time by typing a "leave session" command in window W_c. His user agent process is terminated and a message is broadcast to

all users announcing his departure. The session chairman is not allowed to leave the session before designating another user to be the new chairman. He can do so by typing a "new chairman" command in window W_c. The chairman has the power to dismiss a user from a session (for example, if the user ignores rules or otherwise disrupts the session) by typing a "dismiss user" command. The session can be ended only by the chairman when he types the "end session" command in window W_c. At the end of a session the workspace objects will be copied back to their original places, and the new objects created during the session will be saved under the names provided by the session chairman.

2.8 Session Status and Information

Any user can find out about the current sessions on any given machine by issuing a "list session" command to the communications server CS of his local machine. CS will ask the communications server of the remote machine to provide all information about the ongoing non-secret sessions. This information includes: session-id, participants, chairman, observers, workspace objects, current tool, how long the session has started.

3 Prototype Implementation

We have implemented a prototype of the Remote Shared Workspace model described in Section 2. The programs are written in the C language under Berkeley UNIX version 4.3. The communication between all processes in the same machine or in different machines is based on Berkeley Interprocess Communications facilities. For a complete description of these facilities see [11] and for a tutorial see [12]. We have used the *stream sockets* in the *inet domain* to establish a reliable, bidirectional virtual circuit between each pair of communicating processes, for example between a session server and a user agent.

At this time, we have implemented the static type of session, where the number of participants is predetermined, and therefore the session chairman functions are now limited to starting the session, ending the session and invoking a tool. Each participant creates his windows W_t and W_c using any available window management system (such as the *SUN* windows, *X-windows*, or the 4.3 *window* program that can be used to create windows on any ASCII terminal). After creating the two windows, the user "manually" connects the two windows to the user agent process. Soon we will let the program perform these tasks "automaticaly".

Since a virtual circuit between the session server process and the user agent process carries messages of different sources and meanings, messages are packaged in *frames*. Each frame has three fields: *Type*, *Length* and *Data*. We have the following four types of frames:

token: for token control, e.g., "get token".

session: for session control,e.g.,"start session".

chat: for conversation messages.

tool: for input to and output from the tool process.

The *Length* field has the number of bytes contained in the *Data* field of the frame.

Processes use the *SELECT* system call of Berkeley 4.3BSD to multiplex input from several sources. For example, in Figure 3, process SS receives the input from all user agent processes as well as the tool agent process. For more details about the prototype implementation see [10].

4 Conclusions and Future Goals

We have described a system that enables users on different machines of a computer network to collaborate in real-time using familiar single-user tools. Our system can be viewed as a layer of software to convert single user applications into tools for use by a team of remote users. No modification or restriction to the use of these tools are imposed, the exact commands of the tools as described in their respective manuals can be used during a collaborative session. The following is a list of the major research issues currently under investigation:

1. **Additional types of objects and input devices:** The current implementation deals only with textual objects, and tools are required to accept input from a keyboard. Our next goal is to deal with other types of objects, such as graphics, and other input devices, such as mice.

2. **Object locking:** If an object is used in one session, it may be made available to other sessions in "read only" mode, but not in "read-write" mode. Therefore a "locking" mechanism is needed. If the operating system does not provide it, then locking may be implemented as part of the communications server, or as a separate component of the system.

3. **Object and tool access permissions:** How can users access objects and execute tools on machines for which they have no accounts? As described earlier, remote objects and tools are normally accessed through the remote communications server; therefore it is natural to implement an access and

protection scheme as part of the communications servers.

4. **Performance analysis and evaluation:** The prototype has mainly been used between machines on the same regional network (the MCNC network, connecting the Microelectronics Center of North Carolina to universities such as UNC, NCSU and Duke). Using our system with text editors such as *vi*, response over the network is almost as fast as it is in single-user mode. However, users connected to the session server using slower links than normal (e.g., the ARPANET at daytime) may not be able to keep up and interact with others in a timely fashion. A possible solution to this problem is to require that users connected via slow links have a local copy of the objects and the tool process. In such case only the keystrokes need to travel through the network, and the voluminous output from the tool will be displayed locally. The key to the success of this scheme is to ensure that copies are mutually consistent across the network.

5. **Reliability and Recovery:** Reliability and recovery from errors are very important issues in a distributed environment. For example, what happens if a participant, or the session chairman, is disconnected due to a site or communications failure? Can the system recover if any critical process such as the session server, or the communications server, fails or is not reachable? We are investigating various aspects of the system fault tolerance and recovery.

Acknowledgements

We are grateful to Peter Calingaert, John Menges, John Smith and Michael Stumm for helpful comments. This work was partly supported by ONR under contract N00014-86-K-0680.

REFERENCES

1. I. Greif, 'Computer Support for Cooperative Office Activities', *Proceedings of the 1982 Office Automation Conference,* San Francisco (April 1982).

2. S.K. Sarin, 'Interactive On-Line Conference', *Ph.D. Thesis, MIT,* Also available as Laboratory for Computer Science Technical Report MIT/LCS/TR-330 (1984).

3. G. Foster, 'Collaborative Systems and Multi-user interfaces', *Ph.D. dissertation, Division of Computer Science,* University of California, Berkeley (1986).

4. S. Sarin, and I. Greif, 'Computer-Based Real-Time Conferences', *IEEE Computer 18,10* Special issue on Computer-Based Multimedia Communications, 33-45 (October 1985).

5. R. Seliger, 'The Design and Implementation of a Distributed Program for Collaborative Editing', *Master Thesis, MIT,* Also available as laboratory for Computer Science Technical Report MIT/LCS/TR-350 (1985).

6. M. Stefik, D. G. Borbrow, S. Lanning, D. Tatar, and G. Foster, 'WYSIWIS revised: Early Experience with Multi-user interfaces', *Proceedings of the conference on Computer-Supported Cooperative Work,* Austin Texas 276-290 (December 1986).

7. M. Stefik, G. Foster, D. G. Borbrow, K. Kahn, S. Lanning, and L. Suchman, 'Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings', *Communications of the ACM 30,1* 32-47 (January 1987).

8. N. Jarrell, and W. Barrett, 'Network-based Systems for Asynchronous Group Communication', *Proceedings of the conference on Computer-Supported Cooperative Work,* Austin Texas, 184-191 (December 1986).

9. K. A. Lantz, 'An Experiment in Integrated Multimedia Conferencing', *Proceedings of the conference on Computer-Supported Cooperative Work,* Austin Texas, 267-275 (December 1986).

10. H. Abdel-Wahab, Sheng-Uei Guan, and J. Nievergelt, 'A Prototype for Remotely Shared Textual Workspaces', Department of Computer Science, University of North Carolina, Chapel-Hill (1987).

11. S.J. Leffler, R.S. Fabry, W.N. Joy, P. Lapsley, S. Miller, and C. Torek, 'An Advanced 4.3BSD Interprocess Communication Tutorial', *Computer System Research Group,* Department of Electrical Engineering and Computer Science, University of California, Berkeley (1986).

12. S. Sechrest, 'An Introductory 4.3BSD Interprocess Communication Tutorial', *Computer System Research Group,* Department of Electrical Engineering and Computer Science, University of California, Berkeley (1986).

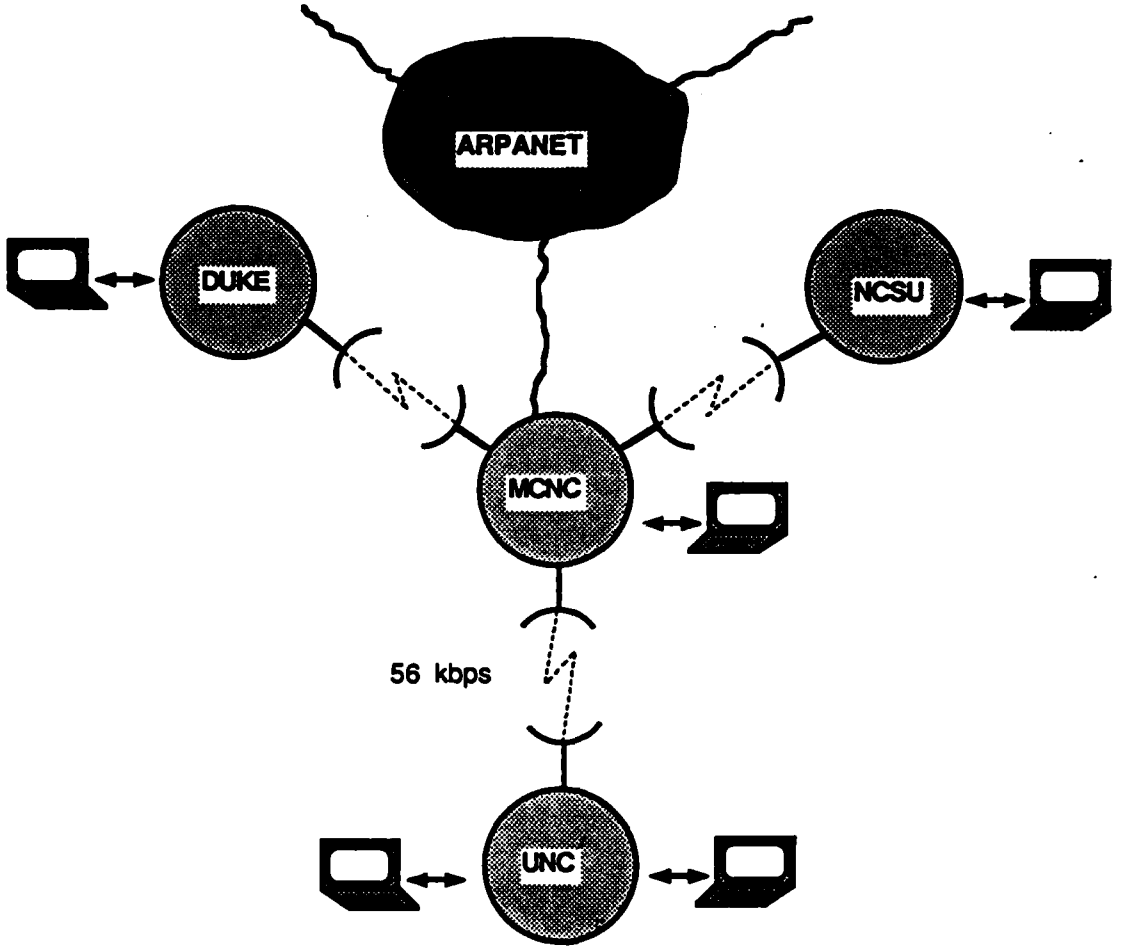


Figure 1: General overview

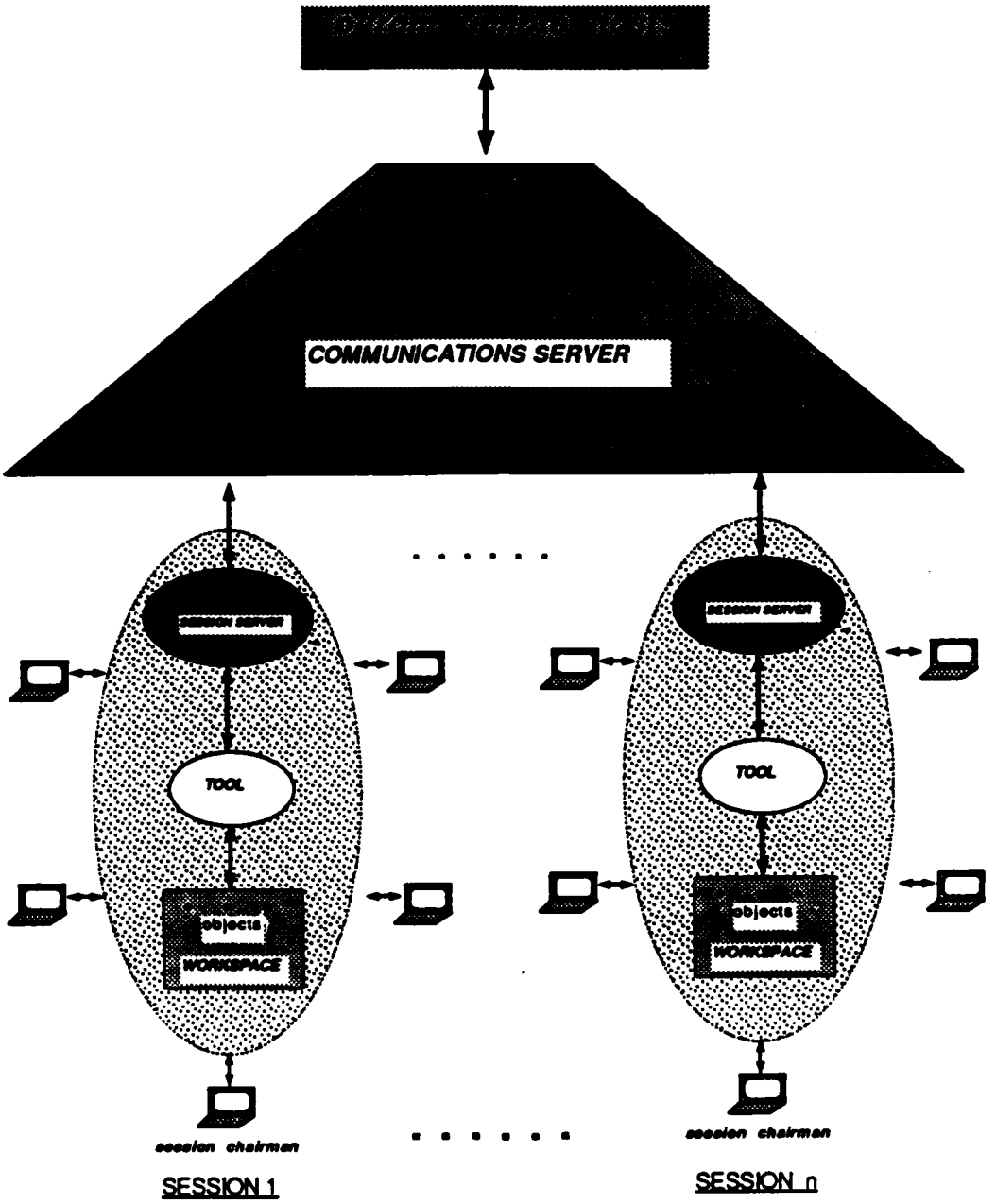


figure 2: system components

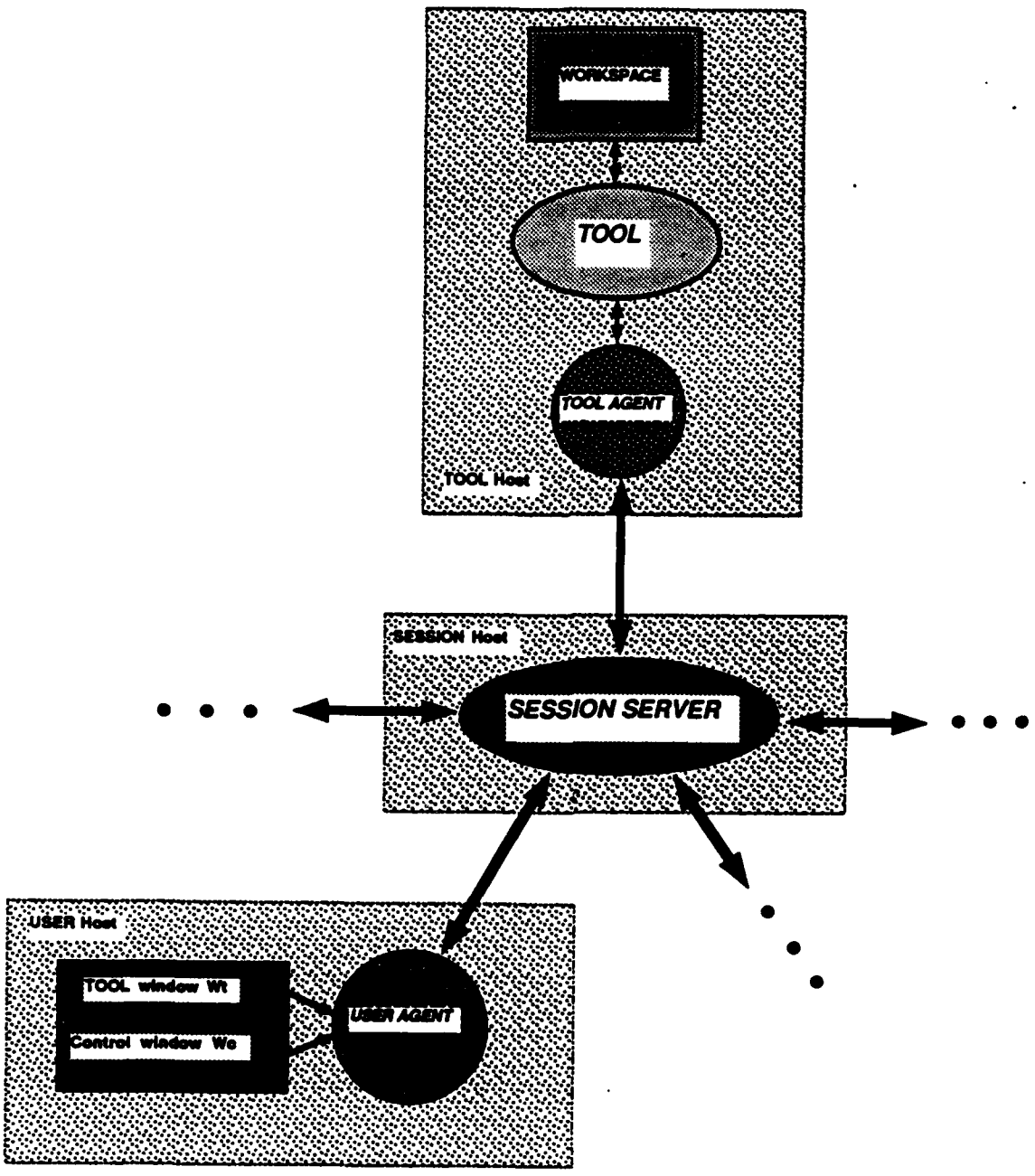


Figure 3: USER and TOOL Agents

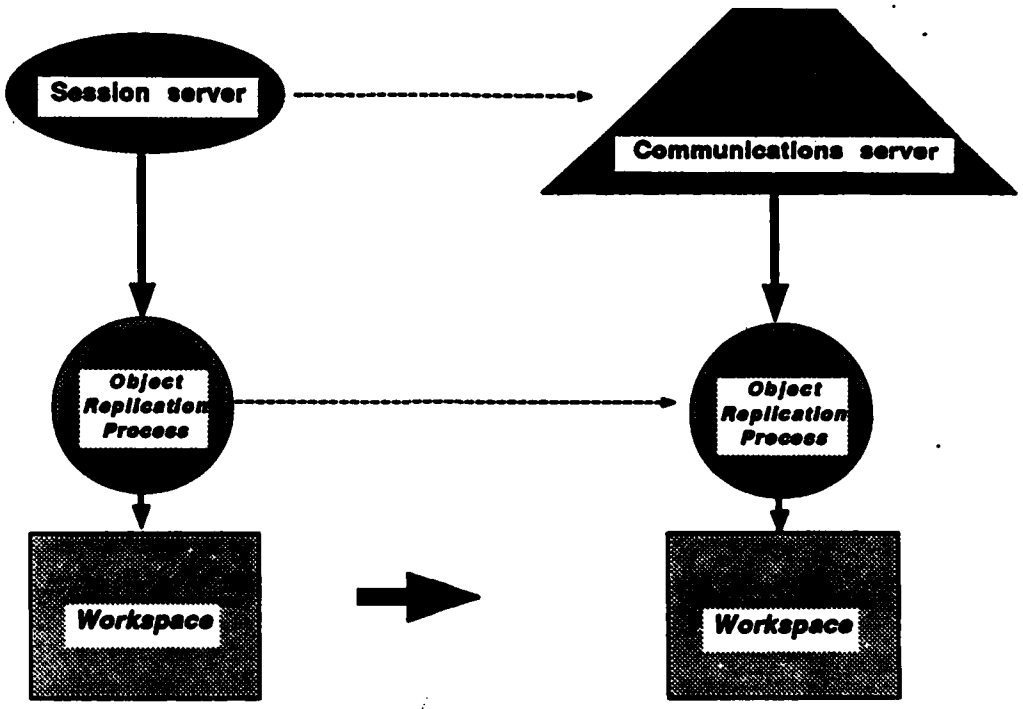


Figure 4: Copying Objects between hosts

END

FILMED

FEB. 1988

DTIC